Università della Calabria

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

---

Corso di Laurea Magistrale in Ingegneria Informatica

Relazione Progetto “Intelligenza Artificiale”

Docenti
Prof. Francesco SCARCELLO
Ing. Antonio BONO

Studenti
Alessandro MANCUSO
Matr. 252050

Anno Accademico 2023/2024

Introduzione

Il presente elaborato è strutturato in tre punti principali:

1. modellazione di un problema di pianificazione classica attraverso il linguaggio PDDL;
2. sviluppo e implementazione di un algoritmo di ricerca e della relativa euristica per risolvere specifiche istanze del problema;
3. utilizzo di azioni con durata (*"durative action"*) e implementazione del problema in PlanSys2 con azioni simulate (*"fake action"*).

A ciascuno dei punti corrisponde un capitolo.

Sommario

1 Task 1 – Modelling

1.1 Modellazione del dominio

In questa sezione si tratta la modellazione di un dominio PDDL che simuli un ambiente di produzione industriale, vengono quindi descritti di seguito i diversi tipi di oggetti definiti, che rappresentano le entità, e le azioni da essi commesse.

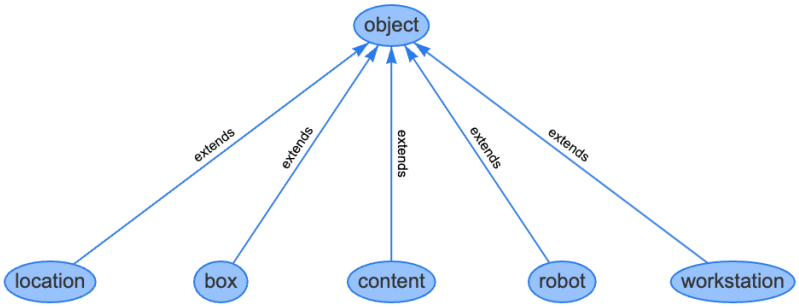
1.1.1 Tipi

Al fine di modellare opportunamente il dominio sono stati definiti i seguenti tipi:

```
;; Tipologie degli oggetti gestiti
Show hierarchy
(:types
    location box content robot workstation
)
```

Dove:

- ***location****:* Rappresenta le posizioni fisiche in cui robot e altri oggetti possono trovarsi.
- ***box***: Scatole utilizzate per trasportare contenuti.
- ***content***: Materiali che possono essere caricati nelle scatole.
- ***robot***: Agenti robotici che effettuano operazioni nel sistema.
- ***workstation***: Postazioni di lavoro in cui i robot possono interagire con contenuti o scatole.



1.1.2 Predicati

Sono stati introdotti dei predicati, che servono a descrivere le proprietà e le relazioni degli oggetti appartenenti a essi, i predicati definiti sono i seguenti:

```
;; Predicati
(:predicates
    (located_at ?entity - (either robot workstation box content) ?loc - location) 15👁 4☑ 4☒
    (is_empty ?box - box) 3👁 2☑ 3☒
    (contains ?box - box ?item - content) 2👁 3☑ 2☒
    (inside_ws ?entity - (either robot box content) ?ws - workstation) 10👁 3☑ 3☒
    (holding ?robot - robot ?box - box) 2👁 2☑ 2☒
    (available ?robot - robot) 7👁 2☑ 2☒ ; Il robot è libero se non sta trasportando nulla
    (linked ?loc1 ?loc2 - location) 1👁
    (is_warehouse ?loc - location) 2👁 ; Identifica il magazzino
)
```

Dove:

- ***located_at***: Indica la posizione corrente di un'entità (robot, workstation, scatola o contenuto).
- ***is_empty***: Determina se una scatola è vuota.
- ***contains***: Specifica se una scatola contiene un determinato contenuto.
- ***inside_ws***: Indica se un'entità si trova all'interno di una workstation.
- ***holding***: Specifica se un robot sta trasportando una scatola.
- ***available***: Indica se un robot è libero (non sta trasportando nulla).
- ***linked***: Definisce la connessione tra due location.
- ***is_warehouse***: Identifica se una location è un magazzino.

1.1.3 Azioni

Le azioni consentono agli agenti di interagire con l'ambiente circostante e di effettuare le operazioni necessarie al soddisfacimento del goal, ogni azione è costituita da: *parametri*, *pre-condizione* ed *effetti*.

Vengono quindi definite le seguenti azioni:

- **move_robot:**

```
; Spostare il robot tra due location adiacenti
(:action move_robot
    :parameters (?r - robot ?from ?to - location)
    :precondition (and (located_at ?r ?from) (linked ?from ?to))
    :effect (and (not (located_at ?r ?from)) (located_at ?r ?to))
)
```

  - **Precondizioni**: Il robot si trova nella location di partenza (?from) e le due location sono collegate.
  - **Effetti**: Il robot viene spostato alla location di destinazione (?to).

- **enter_workstation:**

```
; Il robot entra in una workstation
(:action enter_workstation
    :parameters (?r - robot ?loc - location ?ws - workstation)
    :precondition (and (located_at ?ws ?loc) (located_at ?r ?loc) (not (inside_ws ?r ?ws)))
    :effect (and (not (located_at ?r ?loc)) (inside_ws ?r ?ws))
)
```

  - **Precondizioni**: Il robot e la workstation devono trovarsi nella stessa location.
  - **Effetti**: Il robot entra nella workstation.

- **exit_workstation:**

```
; Il robot esce dalla workstation
(:action exit_workstation
    :parameters (?r - robot ?loc - location ?ws - workstation)
    :precondition (and (located_at ?ws ?loc) (inside_ws ?r ?ws))
    :effect (and (not (inside_ws ?r ?ws)) (located_at ?r ?loc))
)
```

  - **Precondizioni**: Il robot si trova all'interno della workstation.
  - **Effetti**: Il robot esce dalla workstation e ritorna alla location esterna.

- **drop_box_in_workstation:**

```
; Il robot lascia una scatola nella workstation
(:action drop_box_in_ws
    :parameters (?r - robot ?box - box ?ws - workstation)
    :precondition (and (inside_ws ?r ?ws) (holding ?r ?box))
    :effect (and (not (holding ?r ?box)) (available ?r) (inside_ws ?box ?ws))
)
```

  - **Precondizioni**: Il robot è dentro la workstation e sta trasportando una scatola.
  - **Effetti**: La scatola viene lasciata nella workstation e il robot diventa disponibile.

- **drop_box_in_loc:**

```
; Il robot lascia una scatola in una location
(:action drop_box_in_loc
    :parameters (?r - robot ?box - box ?loc - location)
    :precondition (and (located_at ?r ?loc) (holding ?r ?box))
    :effect (and (not (holding ?r ?box)) (available ?r) (located_at ?box ?loc))
)
```

  - **Precondizioni**: Il robot si trova nella location specificata (?loc) e sta trasportando una scatola (?box).
  - **Effetti**: Il robot non trasporta più la scatola (?box), diventa disponibile e la scatola viene lasciata nella location (?loc).

- **pick_box_from_ws:**

```
; Il robot raccoglie una scatola da una workstation
(:action pick_box_from_ws
    :parameters (?r - robot ?box - box ?ws - workstation)
    :precondition (and (inside_ws ?box ?ws) (inside_ws ?r ?ws) (available ?r))
    :effect (and (holding ?r ?box) (not (available ?r)) (not (inside_ws ?box ?ws)))
)
```

  - **Precondizioni**: Il robot si trova all'interno della workstation (?ws), la scatola (?box) è presente nella workstation e il robot è disponibile.
  - **Effetti**: Il robot inizia a trasportare la scatola (?box), non è più disponibile e la scatola viene rimossa dalla workstation (?ws).

- **pick_box_from_loc:**

```
; Il robot raccoglie una scatola da una location
(:action pick_box_from_loc
    :parameters (?r - robot ?box - box ?loc - location)
    :precondition (and (located_at ?box ?loc) (located_at ?r ?loc) (available ?r))
    :effect (and (holding ?r ?box) (not (available ?r)) (not (located_at ?box ?loc)))
)
```

  - **Precondizioni**: Il robot e la scatola (?box) si trovano nella stessa location (?loc) e il robot è disponibile.
  - **Effetti**: Il robot inizia a trasportare la scatola (?box), non è più disponibile e la scatola viene rimossa dalla location (?loc).

- **unload_box_in_ws:**

```
; Il robot svuota una scatola in una workstation
(:action unload_box_in_ws
    :parameters (?r - robot ?box - box ?ws - workstation ?content - content)
    :precondition (and (available ?r) (inside_ws ?r ?ws) (inside_ws ?box ?ws) (contains ?box ?content))
    :effect (and (not (contains ?box ?content)) (is_empty ?box) (inside_ws ?content ?ws))
)
```

  - **Precondizioni**: Il robot si trova all'interno della workstation (?ws) ed è disponibile. La scatola (?box) si trova nella workstation, contiene un contenuto (?content) ed è vuota.
  - **Effetti**: Il contenuto (?content) viene rimosso dalla scatola (?box), che diventa vuota, e il contenuto viene depositato nella workstation (?ws).

- **load_box_in_ws:**

```
; Il robot riempie una scatola in una workstation
(:action load_box_in_ws
    :parameters (?r - robot ?box - box ?ws - workstation ?content - content)
    :precondition (and (available ?r) (inside_ws ?r ?ws) (inside_ws ?box ?ws) (inside_ws ?content ?ws) (is_empty ?box))
    :effect (and (not (inside_ws ?content ?ws)) (contains ?box ?content) (not (is_empty ?box)))
)
```

  - **Precondizioni**: Il robot, la scatola (?box) e il contenuto (?content) si trovano all'interno della workstation (?ws). La scatola è vuota e il robot è disponibile.
  - **Effetti**: Il contenuto (?content) viene messo nella scatola (?box), che non è più vuota.

- **unload_box_in_loc:**

```
; Il robot svuota una scatola in una location
(:action unload_box_in_loc
    :parameters (?r - robot ?box - box ?content - content ?loc - location)
    :precondition (and (available ?r) (located_at ?r ?loc) (located_at ?box ?loc) (contains ?box ?content))
    :effect (and (not (contains ?box ?content)) (is_empty ?box) (located_at ?content ?loc))
)
```

  - **Precondizioni**: Il robot si trova nella location (?loc) ed è disponibile. La scatola (?box) si trova nella location, contiene un contenuto (?content) ed è vuota.
  - **Effetti**: Il contenuto (?content) viene rimosso dalla scatola (?box), che diventa vuota, e il contenuto viene depositato nella location (?loc).

- **load_box_in_loc:**

```
; Il robot riempie una scatola in una location
(:action load_box_in_loc
    :parameters (?r - robot ?box - box ?content - content ?loc - location)
    :precondition (and (not (is_warehouse ?loc)) (available ?r) (located_at ?r ?loc) (located_at ?box ?loc) (located_at ?content ?loc) (is_empty ?box))
    :effect (and (not (located_at ?content ?loc)) (contains ?box ?content) (not (is_empty ?box)))
)
```

  - **Precondizioni**: Il robot, la scatola (?box) e il contenuto (?content) si trovano nella stessa location (?loc). La scatola è vuota e il robot è disponibile.
  - **Effetti**: Il contenuto (?content) viene messo nella scatola (?box), che non è più vuota.

- **load_box_in_warehouse:**

```
; Il robot riempie una scatola nel magazzino
(:action load_box_in_warehouse
    :parameters (?r - robot ?box - box ?content - content ?loc - location)
    :precondition (and (is_warehouse ?loc) (available ?r) (located_at ?r ?loc) (located_at ?box ?loc) (located_at ?content ?loc) (is_empty ?box))
    :effect (and (contains ?box ?content) (not (is_empty ?box)))
)
```

- **Precondizioni**: Il robot, la scatola (`?box`) e il contenuto (`?content`) si trovano nel magazzino (`?loc`). La scatola è vuota, il magazzino è identificato e il robot è disponibile.
- **Effetti**: Il contenuto (`?content`) viene messo nella scatola (`?box`), che non è più vuota.

2 Task 2 – Classical Planning

In questa sezione l'attenzione verrà posta sullo sviluppo di un planner personalizzato (utilizzando il framework PDDL4J) per la risoluzione delle due istanze fornite.

2.1 Istanza 1

Per la prima delle due istanze si fa riferimento al dominio descritto nella precedente sezione, in particolare lo stato iniziale per tale istanza è riportato di seguito:

- Tutte le scatole (*box*) sono collocate in un'unica location, detta *warehouse.*
- Tutti i *content*, che dovranno essere caricati nelle scatole, si trovano nella warehouse.
- Un singolo *agente robotico* è posizionato nella warehouse con il compito di consegnare le scatole.
- Nella warehouse non sono presenti *workstation*.

L'obiettivo (il *goal*) è quello di consegnare tutte le scatole alle workstation in base alle loro richieste.

Si sottolinea che non viene fornita alcun tipo di restrizione sul numero di scatole da utilizzare, nel caso in esame vengono utilizzate 3 scatole.

L'istanza che si è scelta di utilizzare è la seguente:

```
(define (problem manufacturing_scenario)
  (:domain manufacturing)

  ;; Oggetti presenti nello scenario
  Show hierarchy
  (:objects
    robot - robot
    box1 box2 - box
    bolts valves - content
    warehouse station1 station2 - location
    ws1 ws2 - workstation
  )
```

Gli oggetti inizializzati sono i seguenti:

- ***Robot**:*
  - *robot:* Il robot responsabile della movimentazione delle scatole.
- ***Scatole (Box)**:*
  - *box1 e box2:* Scatole utilizzate per trasportare contenuti.
- ***Contenuti (Content)**:*
  - *bolts:* Bulloni da consegnare.
  - *valves:* Valvole da consegnare.

- ***Location**:*
  - *warehouse:* Il magazzino centrale.
  - *station1, station2:* Stazioni intermedie o punti di trasferimento.
- ***Workstation**:*
  - *ws1 e ws2:* Workstation che devono ricevere i contenuti.

```
;; Stato iniziale del sistema
View
(:init
  ;; Posizione iniziale degli oggetti
  (located_at robot warehouse)

  (located_at box1 warehouse)
  (located_at box2 warehouse)

  (located_at bolts warehouse)
  (located_at valves warehouse)

  (is_empty box1)
  (is_empty box2)

  (available robot)

  ;; Connessioni tra location
  (linked warehouse station1)
  (linked station1 station2)
  (linked station2 warehouse)

  ;; Identificazione del magazzino
  (is_warehouse warehouse)
)
```

Lo stato iniziale specifica che tutti gli oggetti si trovano nel magazzino e che il robot è pronto per eseguire operazioni.

```
  ;; Obiettivi da raggiungere
  (:goal
    (and
        (inside_ws box1 ws1)
        (inside_ws box2 ws2)

        (contains box1 bolts)
        (contains box2 valves)

    )
  )
)
```

L'obiettivo (il **goal**) consiste nell'avere oggetti di tipo *bolts* nella *workstation1* e oggetti di tipo *valves* nella *workstation2*.

2.2 Istanza 2

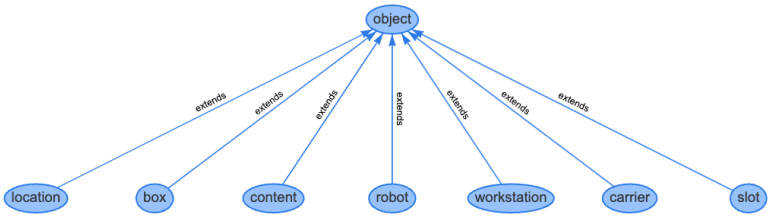
2.2.1 Dominio

Per descrivere correttamente il dominio della seconda istanza bisogna tenere conto delle estensioni introdotte, quali:

- Ogni agent è dotato di un unico **carrier** con una capacità di carico massima che può variare.
- Ogni carrier è dotato di uno o più *slot* che le box possono occupare durante il trasporto.
- Per ciascun agent viene monitorato:
    - Quali box sono state caricate sul carrier.
    - Numero totale delle box sul carrier.

A tal fine rispetto all'*Istanza 1*, sono stati modificati tipi e predicati, aggiungendo i tipi *carrier* e *slot*.

```
;; Tipologie degli oggetti gestiti
Show hierarchy
(:types
    location box content robot workstation carrier slot
)
```



I predicati utilizzati sono i seguenti:

```
;; Predicati
(:predicates
    (located_at ?entity - (either robot workstation box content carrier) ?loc - location) 17 5 5
    (is_empty ?box - box) 3 2 3
    (contains ?box - box ?item - content) 2 3 2
    (inside_ws ?entity - (either robot box content) ?ws - workstation) 10 3 3
    (carrying ?carrier - carrier ?box - box) 2 2 2    ; Si utiilzza Carrier invece del Robot
    ;(available ?robot - robot)
    (linked ?loc1 ?loc2 - location) 2
    (is_warehouse ?loc - location) 4 ; Identifica il magazzino

    ; NUOVI PREDICATI introdotti per soddisfare le richieste
    (joined ?robot - robot ?carrier - carrier) 13
    (handle ?carrier - carrier ?slot - slot) 5 ; Il carrier possiede questo slot
    (available ?slot - slot) 3 2 2 ; Uno slot è libero per essere popolato
)
```

Si può notare come il predicato *aviable/1* viene rimosso in quanto, utilizzando il carrier non vi è più la necessità di preoccuparsi dello stato del robot, verrà riutilizzato successivamente per verificare la disponibilità o meno di uno slot all'interno del carrier.

I predicati aggiunti sono:

- **joined**: Specifica che un robot è associato a un carrier.
- **handle**: Descrive la relazione tra un carrier e i suoi slot.
- **available**: Verifica se uno slot è libero e può essere utilizzato.

Al fine di adattare il nuovo dominio, sono state effettuate modifiche alle azioni precedentemente definite, mantenendo però intatta la struttura precedentemente descritta.

```
; Spostare il robot tra due location adiacenti (che non sia la warehouse)
(:action move_robot
    :parameters (?r - robot ?from ?to - location ?c - carrier)
    :precondition (and (located_at ?r ?from) (linked ?from ?to) (not (is_warehouse ?to)) (joined ?r ?c))
    :effect (and (not (located_at ?r ?from)) (located_at ?r ?to))
)

; Spostare il robot tra due location adiacenti (che sia la warehouse)
(:action move_robot_warehouse
    :parameters (?r - robot ?from ?to - location ?c - carrier)
    :precondition (and (located_at ?r ?from) (linked ?from ?to) (is_warehouse ?to) (joined ?r ?c)
        (forall (?slot - slot)
            (and (handle ?c ?slot) (available ?slot))
        )
    )
    :effect (and (not (located_at ?r ?from)) (located_at ?r ?to))
    ; Vi è la necessità di differenziare queste azioni in quanto nella warehouse si può accedere solo con il carrier vuoto
)
```

È stata aggiunta l'azione **move_robot_warehouse**, presentando la necessità di differenziare rispetto all'azione **move_robot**, in quanto dal momento in cui si utilizzano i *carrier* si potrà accedere alla warehouse unicamente con il carrier vuoto, si può osservare infatti come nelle precondizioni di questa azione si verifica che tutti gli slot siano disponibili.

2.2.2 Problema

Per il problema relativo all'Istanza2 è stata utilizzata la seguente istanza:

```
(:objects
  robot - robot
  carrier - carrier
  warehouse location1 location2 - location
  workstation1 workstation2 workstation3 - workstation
  box1 box2 box3 - box
  bolt valve - content
  slot1 slot2 slot3 - slot    ; Il carrier ha quindi capacità pari a 3 slot
)
```

Gli oggetti definiti sono i seguenti:

- **Robot**:
    - *robot*: L'agente responsabile della movimentazione del carrier.
- **Carrier**:
    - *carrier*: Contenitore mobile utilizzato dal robot per trasportare scatole.

- **Location**:
  - *warehouse*: Magazzino centrale.
  - *location1, location2*: Altre location nel sistema.
- **Workstation**:
  - *workstation1, workstation2, workstation3*: Postazioni di lavoro che necessitano scatole e contenuti.
- **Box**:
  - *box1, box2, box3:* Scatole utilizzate per contenere e trasportare contenuti.
- **Content**:
  - *bolt, valve:* Contenuti da trasportare.
- **Slot**:
  - s*lot1, slot2, slot3*: Spazi nel carrier che rappresentano la capacità massima (3 slot).

```
(:init
  ; Il robot si trova nella warehouse
  (located_at robot warehouse)

  ; Colleghiamo il carrier al robot
  (joined robot carrier)

  ; Identifichiamo la warehouse
  (is_warehouse warehouse)

  ; I box sono collocati nella warehouse
  (located_at box1 warehouse)
  (located_at box2 warehouse)
  (located_at box3 warehouse)

  ; I contenuti da mettere nei box sono nella warehouse
  (located_at bolt warehouse)
  (located_at valve warehouse)

  ; Le scatole sono vuote
  (is_empty box1)
  (is_empty box2)
  (is_empty box3)

  ; Posizioniamo le workstation
  (located_at workstation1 location1)
  (located_at workstation2 location2)
  (located_at workstation3 location2)

  ; Connessioni
  (linked warehouse location1)
  (linked location1 location2)
  (linked location1 warehouse)
  (linked location2 location1)

  ; Impostiamo la capacità del carrier
  (handle carrier slot1)
  (handle carrier slot2)
  (handle carrier slot3)
  (available slot1)
  (available slot2)
  (available slot3)
)
```

Lo stato iniziale è molto simile a quello dell'Istanza1, adattata ovviamente alle nuove richieste.

```
(:goal
  (and
    ; Almeno una workstation che necessita di un bullone
    (inside_ws bolt workstation2)

    ; Almeno una workstation che non necessita di nulla
    (not (inside_ws bolt workstation1))
    (not (inside_ws valve workstation1))

    ; Almeno una workstation che necessita di bulloni e viti
    (inside_ws bolt workstation3)
    (inside_ws valve workstation3)
  )
)
```

L'obiettivo (**goal**) consiste nel soddisfare che almeno una workstation abbia un oggetto di tipo *bolt*, almeno una workstation abbia sia un oggetto di tipo *bolt* che di tipo *valve*, e che almeno una workstation non necessiti di alcun oggetto.

Dopo aver definito le istanza, si passa alla spiegazione del planner implementato.

2.3 Planner Details

In questa sezione viene fornita una descrizione dettagliata del planner implementato mediante il framework *PDDL4J.*

Il planner è stato implementato facendo riferimento alla guida presente nella documentazione ufficiale di *PDDL4J* presso il seguente indirizzo web:

http://pddl4j.imag.fr/writing_your_own_planner.html.

In particolare, si ha una classe chiamata *ASP* che estende la classe astratta *AbstractPlanner*.

La classe base è stata modificata con l'aggiunta di un campo *useNewHeuristic* il cui scopo è quello di utilizzare l'euristica personalizzata che verrà descritta successivamente.

Dato questo campo, sono stati implementati i metodi *getter* e *setter*, dove in quest'ultimo caso bisogna sottolineare la seguente implementazione:

```
//Metodo per settare la nuova euristica
// 0 -> Si usa l'algorimto di ricerca con un eristica di base
// 1 -> Si usa l'algoritmo di ricerca con l'euristica definita ad hoc
@CommandLine.Option(names = {"-en", "--heuristicNew"}, defaultValue = "0",
        description = "Set the heuristic : 1")
public void setHeuristicNew(int heuristicsNew) {
    this.useNewHeuristic = heuristicsNew;
}
```

Vi è quindi la possibilità di poter definire direttamente da linea di comando quale euristica utilizzare, in particolare:

- **en 0:** si usa l'algoritmo di ricerca con un *euristica di base* definita con il comando -n;
- **en 1:** si usa l'algoritmo di ricerca con *l'euristica custom*, in questo caso ovviamente non è necessario inserire da riga di comando -n.

Il planner sfrutta l'algoritmo di ricerca A*, in particolare si utilizza una versione modificata ad hoc di A* al fine di utilizzare l'euristica proposta, di seguito viene riportato il codice di *customAStar*:

```
//Algoritmo A* modificato ad hoc (definito come nella guida di PDDL4J ma con l'euristica custom)
    public Plan customAstar(Problem problem){

        //Costruiamo la nostra euristica
        CustomHeuristic heuristic = new CustomHeuristic(problem);

        // Ci prendiamo lo stato iniziale
        final State init = new State(problem.getInitialState());

        // Inizializziamo l'insieme dei nodi visitati
        final Set<Node> close = new HashSet<>();


        final double weight = this.getHeuristicWeight();
        final PriorityQueue<Node> open = new PriorityQueue<>(100, new Comparator<Node>() {
            public int compare(Node n1, Node n2) {
                double f1 = weight * n1.getHeuristic() + n1.getCost();
                double f2 = weight * n2.getHeuristic() + n2.getCost();
                return Double.compare(f1, f2);
            }
        });

        // Definiamo la radice dell'albero di ricerca
        final Node root = new Node(init, null, -1, 0, heuristic.estimate(init, problem.getGoal()));


        open.add(root);
        Plan plan = null;

        // Settiamo il timeout per la ricerca
        final int timeout = this.getTimeout() * 1000;
        long time = 0;

        // Inizio ricerca
        while (!open.isEmpty() && plan == null && time < timeout) {

            // Aggiungiamo il nodo a quelli visitati
            final Node current = open.poll();
            close.add(current);

            //Se il goal è soddisfatto allora restituiamo il plan
            if (current.satisfy(problem.getGoal())) {
                return this.extractPlan(current, problem);
            } else {
                //Altrimenti proviamo ad applicare le azioni al nodo
                for (int i = 0; i < problem.getActions().size(); i++) {
                    // Ci prendiamo le azioni
```

```
                    Action a = problem.getActions().get(i);
                    // Verifichiamo se l'azione è applicabile al problema
                    if (a.isApplicable(current)) {
                        Node next = new Node(current);
                        // Applichiamo l'effetto dell'azione
                        final List<ConditionalEffect> effects = a.getConditionalEffects();
                        for (ConditionalEffect ce : effects) {
                            if (current.satisfy(ce.getCondition())) {
                                next.apply(ce.getEffect());
                            }
                        }
                        //Andiamo ad inserire le informazioni al nuovo nodo
                        final double g = current.getCost() + 1;
                        if (!close.contains(next)) {
                            next.setCost(g);
                            next.setParent(current);
                            next.setAction(i);
                            next.setHeuristic(heuristic.customEstimate(next, problem.getGoal(), a,
                                              current.getHeuristic()));
                            open.add(next);
                        }
                    }
                }
            }
        }
        //Restituiamo il plan -> Null se vuoto
        return plan;
    }
```

Il metodo principale della classe è il metodo *solve*, viene riportato di seguito il suo codice:

```
//Metodo principale della classe
    @Override
    public Plan solve(final Problem problem) throws ProblemNotSupportedException {
        //Andiamo a vedere per prima cosa se il problema è risolvibile dal plan costruito
        if (!this.isSupported(problem)) { throw new ProblemNotSupportedException("Cannot solve the
problem");}

        //ASTAR
        //mediante la seguente variabile di classe andiamo a scegliere l'euristica di riferimento
        if (this.getHeuristicNew() == 1){

            LOGGER.info("* Starting A* Search with Custom Heuristic \n");

                //Per poter tenere traccia della memoria usata dall'algoritmo
                MemoryMXBean memoryBean = ManagementFactory.getMemoryMXBean();
                MemoryUsage beforeHeapMemoryUsage = memoryBean.getHeapMemoryUsage();
                long beforeUsedMemory = beforeHeapMemoryUsage.getUsed();

                //Per tenere traccia del tempo usato dall'algoritmo
                final long begin = System.currentTimeMillis();
```

```
            //Lanciamo l'algoritmo
            final Plan plan = this.customAstar(problem);
            final long end = System.currentTimeMillis();

            MemoryUsage afterHeapMemoryUsage = memoryBean.getHeapMemoryUsage();
            long afterUsedMemory = afterHeapMemoryUsage.getUsed();

            long memoryUsedByCustomAstar = afterUsedMemory - beforeUsedMemory;

            if (plan != null) {
                LOGGER.info("* A* search succeeded\n");
                this.getStatistics().setTimeToSearch(end - begin);
                this.getStatistics().setMemoryUsedToSearch(memoryUsedByCustomAstar);
            } else {
                LOGGER.info("* A* search failed\n");
            }
            return plan;
        }
        else{
            //In questo caso usiamo l'algoritmo A* con una delle euristiche fornite dal
              framework
            LOGGER.info("* Starting A* search with heuristic: "+ this.getHeuristic()+"\n");
            StateSpaceSearch search = StateSpaceSearch.getInstance(SearchStrategy.Name.ASTAR,
                    this.getHeuristic(), this.getHeuristicWeight(), this.getTimeout());
            LOGGER.info("* Starting A* search \n");
            // Cerchiamo una soluzione
            Plan plan = search.searchPlan(problem);

            if (plan != null) {
                LOGGER.info("* A* search succeeded\n");
                this.getStatistics().setTimeToSearch(search.getSearchingTime());
                this.getStatistics().setMemoryUsedToSearch(search.getMemoryUsed());
            } else {
                LOGGER.info("* A* search failed\n");
            }

            return plan;
        }
    }
```

Nel metodo *solve* viene mostrato come avviene l'utilizzo del campo di classe *useNewHeuristic* introdotto precedentemente, rispetto la scelta dell'euristica da utilizzare in combinata all'algoritmo A*.

2.4 Euristiche Utilizzate

Le funzioni euristiche utilizzate le risolvere le istanze precedentemente esposte sono le seguenti:

- **Fast-Forward:** L'euristica FastForward, spesso abbreviata come FF, è un'euristica basata sui piani rilassati. È stata introdotta da J. Hoffmann e deriva dall'algoritmo Graphplan, il quale provvede alla costruzione di un grafo a livelli alternati di proposizioni e azioni. Tale grafo serve a rappresentare le possibili evoluzioni di uno stato iniziale verso uno stato finale;
- **AdjustedSum:** L'euristica AdjustedSum è un'euristica additiva, proposta da Nguyen and Kambhampati, che tenta di migliorare la stima sommando i costi delle singole sotto-attività, ma con aggiustamenti per tenere conto delle interazioni tra le attività.
- **CustomHeuristic:** Un'euristica costruita ad hoc, di cui successivamente verrà descritto brevemente il codice, che tende a stimare l'euristica dei nodi futuri in base al valore dell'euristica del nodo corrente e dell'azione che bisogna effettuare.

L'euristica custom definita durante lo svolgimento del presente progetto è la seguente:

```
import fr.uga.pddl4j.heuristics.state.RelaxedGraphHeuristic;
import fr.uga.pddl4j.planners.statespace.search.Node;
import fr.uga.pddl4j.problem.Problem;
import fr.uga.pddl4j.problem.State;
import fr.uga.pddl4j.problem.operator.Action;
import fr.uga.pddl4j.problem.operator.Condition;
import fr.uga.pddl4j.problem.Fluent;
import fr.uga.pddl4j.util.BitVector;

public final class CustomHeuristic extends RelaxedGraphHeuristic {

    private final int totalBoxes;
    private final int totalWorkstations;
    private Problem problem;

    public CustomHeuristic(Problem problem) {
        super(problem);
        super.setAdmissible(false); // Non ammissibile per favorire performance

        // Conta il numero totale di scatole e workstation basandosi sui fluents
        this.totalBoxes = countFluentsByType(problem, "box");
        this.totalWorkstations = countFluentsByType(problem, "workstation");
        this.problem = problem;
    }

    @Override
    public int estimate(State state, Condition goal) {
        super.setGoal(goal);
```

```
        super.expandRelaxedPlanningGraph(state);

        // Ottieni la capacità del carrier dinamicamente
        int carrierCapacity = getCarrierCapacity(state);

        // Stima i viaggi necessari
        int tripsNeeded = (int) Math.ceil((double) totalBoxes / carrierCapacity);

        // Stima la distanza totale tra il magazzino e le workstation
        int distanceEstimate = estimateTotalDistance();

        // Combina le stime
        return tripsNeeded * distanceEstimate;
    }

    @Override
    public double estimate(Node node, Condition goal) {
        return this.estimate((State) node, goal);
    }

    public double customEstimate(State state, Condition goal, Action action, double
currentHeuristic) {
        // Penalità per azioni specifiche
        double actionPenalty = action.getName().contains("move") ? 2.0 : 0.0;
        actionPenalty += action.getName().contains("load") ? 1.0 : 0.0;
        actionPenalty += action.getName().contains("unload") ? 0.5 : 0.0;

        // Combina stima dei viaggi, distanze e penalità
        return estimate(state, goal) + actionPenalty + (0.1 * currentHeuristic);
    }

    private int getCarrierCapacity(State state) {
        int capacity = 0;

        // Scansiona i fluents definiti nel problema per identificare gli slot
        for (Fluent fluent : this.problem.getFluents()) {
            String fluentName = fluent.toString();

            // Verifica se il fluent è di tipo "slot"
            if (fluentName.contains("slot") && fluentName.startsWith("(available")) {
                int fluentIndex = this.problem.getFluents().indexOf(fluent);

                if (fluentIndex != -1) {
                    // Crea un BitVector per rappresentare il fluent
                    BitVector positiveFluents = new BitVector();
                    positiveFluents.set(fluentIndex);

                    // Costruisce la condizione
                    Condition slotCondition = new Condition(positiveFluents, new BitVector());

                    // Verifica se lo slot è disponibile nello stato attuale
```

```
                    if (state.satisfy(slotCondition)) {
                        capacity++;
                    }
                }
            }
        }
        return Math.max(capacity, 1); // Garantisce che la capacità sia almeno 1
    }

    private int estimateTotalDistance() {
        // Stima la distanza totale tra il magazzino e tutte le workstation
        return totalWorkstations; // Esempio: Supponiamo che ogni workstation richieda un costo
unitario
    }

    private int countFluentsByType(Problem problem, String type) {
        int count = 0;

        // Scorri i fluents definiti nel problema
        for (Fluent fluent : problem.getFluents()) {
            String fluentName = fluent.toString();

            // Verifica se il nome del fluent corrisponde al tipo richiesto
            if (fluentName.contains(type)) {
                count++;
            }
        }

        return count;
    }
}
```

Si può quindi notare che *CustomHeuristic* estende la classe *RelaxedGraphHeuristic* di PDDL4J, ed è progettata per essere un'euristica *non ammissibile* da utilizzare nell'algoritmo di ricerca A*. L'obiettivo dell'euristica è fornire una stima del costo necessario per raggiungere uno stato obiettivo a partire da uno stato attuale, tenendo in considerazione specifiche caratteristiche del problema, come:

1. Capacità **dinamica** del *carrier*.
2. Numero totale di *scatole* e *workstation*.
3. *Penalità* associate a particolari azioni (come *move*, *load* e *unload*).
4. *Viaggi* necessari e *distanze* complessive.

2.5 Risultati Ottenuti

In questa sezione verranno mostrati tutti i plan ottenuti con l'algoritmo A* e le rispettive euristiche descritte precedentemente.

2.5.1 Istanza 1

Il plan trovato da **Fast-Forward**, utilizzando planutils è il seguente:

FF

```
* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD heuristic
* ENFORCED_HILL_CLIMBING search succeeded

found plan as follows:

00: ( load_box_in_warehouse robot1 box1 bolt warehouse) [0]
01: (          pick_box_from_loc robot1 box1 warehouse) [0]
02: (            move_robot robot1 warehouse location1) [0]
03: (            move_robot robot1 location1 location2) [0]
04: (           enter_workstation robot1 location2 ws2) [0]
05: (                   drop_box_in_ws robot1 box1 ws2) [0]
06: (            unload_box_in_ws robot1 box1 ws2 bolt) [0]
07: (            exit_workstation robot1 location2 ws2) [0]
08: (            move_robot robot1 location2 location1) [0]
09: (            move_robot robot1 location1 warehouse) [0]
10: ( load_box_in_warehouse robot1 box2 bolt warehouse) [0]
11: ( load_box_in_warehouse robot1 box3 bolt warehouse) [0]
12: (          pick_box_from_loc robot1 box2 warehouse) [0]
13: (            move_robot robot1 warehouse location1) [0]
14: (           enter_workstation robot1 location1 ws1) [0]
15: (                   drop_box_in_ws robot1 box2 ws1) [0]
16: (            exit_workstation robot1 location1 ws1) [0]
17: (            move_robot robot1 location1 warehouse) [0]
18: (     unload_box_in_loc robot1 box3 bolt warehouse) [0]
19: (load_box_in_warehouse robot1 box3 valve warehouse) [0]
20: (          pick_box_from_loc robot1 box3 warehouse) [0]
21: (            move_robot robot1 warehouse location1) [0]
22: (           enter_workstation robot1 location1 ws1) [0]
23: (                   drop_box_in_ws robot1 box3 ws1) [0]
24: (            unload_box_in_ws robot1 box2 ws1 bolt) [0]
25: (           unload_box_in_ws robot1 box3 ws1 valve) [0]
26: (                 pick_box_from_ws robot1 box2 ws1) [0]
27: (            exit_workstation robot1 location1 ws1) [0]
28: (            move_robot robot1 location1 warehouse) [0]
29: (            drop_box_in_loc robot1 box2 warehouse) [0]
30: ( load_box_in_warehouse robot1 box2 bolt warehouse) [0]
31: (          pick_box_from_loc robot1 box2 warehouse) [0]
32: (            move_robot robot1 warehouse location1) [0]
33: (            move_robot robot1 location1 location2) [0]
34: (           enter_workstation robot1 location2 ws3) [0]
35: (                   drop_box_in_ws robot1 box2 ws3) [0]
36: (            unload_box_in_ws robot1 box2 ws3 bolt) [0]
37: (                pick_box_from_ws robot1 box2 ws3) [0]
38: (            exit_workstation robot1 location2 ws3) [0]
39: (            move_robot robot1 location2 location1) [0]
40: (            move_robot robot1 location1 warehouse) [0]
41: (            drop_box_in_loc robot1 box2 warehouse) [0]
42: (load_box_in_warehouse robot1 box2 valve warehouse) [0]
43: (          pick_box_from_loc robot1 box2 warehouse) [0]
44: (            move_robot robot1 warehouse location1) [0]
45: (            move_robot robot1 location1 location2) [0]
46: (           enter_workstation robot1 location2 ws3) [0]
47: (                   drop_box_in_ws robot1 box2 ws3) [0]
48: (           unload_box_in_ws robot1 box2 ws3 valve) [0]

time spent:       0,02 seconds parsing
                  0,03 seconds encoding
                  0,51 seconds searching
                  0,56 seconds total time

memory used:      0,51 MBytes for problem representation
                  0,00 MBytes for searching
                  0,52 MBytes total
```

A* con euristica FF

```
* Starting A* search with heuristic: FAST_FORWARD
* Starting A* search
* A* search succeeded

found plan as follows:

00: (load_box_in_warehouse robot1 box2 valve warehouse) [0]
01: ( load_box_in_warehouse robot1 box3 bolt warehouse) [0]
02: (load_box_in_warehouse robot1 box1 valve warehouse) [0]
03: (          pick_box_from_loc robot1 box1 warehouse) [0]
04: (            move_robot robot1 warehouse location1) [0]
05: (           enter_workstation robot1 location1 ws1) [0]
06: (                   drop_box_in_ws robot1 box1 ws1) [0]
07: (           unload_box_in_ws robot1 box1 ws1 valve) [0]
08: (                 pick_box_from_ws robot1 box1 ws1) [0]
09: (            exit_workstation robot1 location1 ws1) [0]
10: (            move_robot robot1 location1 warehouse) [0]
11: (            drop_box_in_loc robot1 box1 warehouse) [0]
12: ( load_box_in_warehouse robot1 box1 bolt warehouse) [0]
13: (          pick_box_from_loc robot1 box1 warehouse) [0]
14: (            move_robot robot1 warehouse location1) [0]
15: (           enter_workstation robot1 location1 ws1) [0]
16: (                   drop_box_in_ws robot1 box1 ws1) [0]
17: (            unload_box_in_ws robot1 box1 ws1 bolt) [0]
18: (                 pick_box_from_ws robot1 box1 ws1) [0]
19: (            exit_workstation robot1 location1 ws1) [0]
20: (            move_robot robot1 location1 warehouse) [0]
21: (            drop_box_in_loc robot1 box1 warehouse) [0]
22: ( load_box_in_warehouse robot1 box1 bolt warehouse) [0]
23: (          pick_box_from_loc robot1 box2 warehouse) [0]
24: (            move_robot robot1 warehouse location1) [0]
25: (            move_robot robot1 location1 location2) [0]
26: (           enter_workstation robot1 location2 ws3) [0]
27: (                   drop_box_in_ws robot1 box2 ws3) [0]
28: (            exit_workstation robot1 location2 ws3) [0]
29: (            move_robot robot1 location2 location1) [0]
30: (            move_robot robot1 location1 warehouse) [0]
31: (          pick_box_from_loc robot1 box1 warehouse) [0]
32: (            move_robot robot1 warehouse location1) [0]
33: (            move_robot robot1 location1 location2) [0]
34: (           enter_workstation robot1 location2 ws2) [0]
35: (                   drop_box_in_ws robot1 box1 ws2) [0]
36: (            unload_box_in_ws robot1 box1 ws2 bolt) [0]
37: (            exit_workstation robot1 location2 ws2) [0]
38: (            move_robot robot1 location2 location1) [0]
39: (            move_robot robot1 location1 warehouse) [0]
40: (          pick_box_from_loc robot1 box3 warehouse) [0]
41: (            move_robot robot1 warehouse location1) [0]
42: (            move_robot robot1 location1 location2) [0]
43: (           enter_workstation robot1 location2 ws3) [0]
44: (                   drop_box_in_ws robot1 box3 ws3) [0]
45: (           unload_box_in_ws robot1 box2 ws3 valve) [0]
46: (            unload_box_in_ws robot1 box3 ws3 bolt) [0]

time spent:       0,02 seconds parsing
                  0,03 seconds encoding
                 36,07 seconds searching
                 36,11 seconds total time

memory used:      0,51 MBytes for problem representation
                488,37 MBytes for searching
                488,88 MBytes total
```

A* con euristica AS	A* con euristica Custom
* Starting A* search with heuristic: AJUSTED_SUM * Starting A* search * A* search succeeded found plan as follows: 00: (load_box_in_warehouse robot1 box3 bolt warehouse) [0] 01: (pick_box_from_loc robot1 box3 warehouse) [0] 02: (move_robot robot1 warehouse location1) [0] 03: (move_robot robot1 location1 location2) [0] 04: (enter_workstation robot1 location2 ws2) [0] 05: (drop_box_in_ws robot1 box3 ws2) [0] 06: (unload_box_in_ws robot1 box3 ws2 bolt) [0] 07: (pick_box_from_ws robot1 box3 ws2) [0] 08: (exit_workstation robot1 location2 ws2) [0] 09: (move_robot robot1 location2 location1) [0] 10: (move_robot robot1 location1 warehouse) [0] 11: (drop_box_in_loc robot1 box3 warehouse) [0] 12: (load_box_in_warehouse robot1 box3 valve warehouse) [0] 13: (load_box_in_warehouse robot1 box2 valve warehouse) [0] 14: (load_box_in_warehouse robot1 box1 bolt warehouse) [0] 15: (pick_box_from_loc robot1 box1 warehouse) [0] 16: (move_robot robot1 warehouse location1) [0] 17: (enter_workstation robot1 location1 ws1) [0] 18: (drop_box_in_ws robot1 box1 ws1) [0] 19: (unload_box_in_ws robot1 box1 ws1 bolt) [0] 20: (pick_box_from_ws robot1 box1 ws1) [0] 21: (exit_workstation robot1 location1 ws1) [0] 22: (move_robot robot1 location1 warehouse) [0] 23: (drop_box_in_loc robot1 box1 warehouse) [0] 24: (load_box_in_warehouse robot1 box1 bolt warehouse) [0] 25: (pick_box_from_loc robot1 box2 warehouse) [0] 26: (move_robot robot1 warehouse location1) [0] 27: (move_robot robot1 location1 location2) [0] 28: (enter_workstation robot1 location2 ws3) [0] 29: (drop_box_in_ws robot1 box2 ws3) [0] 30: (unload_box_in_ws robot1 box2 ws3 valve) [0] 31: (exit_workstation robot1 location2 ws3) [0] 32: (move_robot robot1 location2 location1) [0] 33: (move_robot robot1 location1 warehouse) [0] 34: (pick_box_from_loc robot1 box3 warehouse) [0] 35: (move_robot robot1 warehouse location1) [0] 36: (enter_workstation robot1 location1 ws1) [0] 37: (drop_box_in_ws robot1 box3 ws1) [0] 38: (unload_box_in_ws robot1 box3 ws1 valve) [0] 39: (exit_workstation robot1 location1 ws1) [0] 40: (move_robot robot1 location1 warehouse) [0] 41: (pick_box_from_loc robot1 box1 warehouse) [0] 42: (move_robot robot1 warehouse location1) [0] 43: (move_robot robot1 location1 location2) [0] 44: (enter_workstation robot1 location2 ws3) [0] 45: (drop_box_in_ws robot1 box1 ws3) [0] 46: (unload_box_in_ws robot1 box1 ws3 bolt) [0] time spent: 0,02 seconds parsing 0,03 seconds encoding 22,99 seconds searching 23,04 seconds total time memory used: 0,51 MBytes for problem representation 390,25 MBytes for searching 390,76 MBytes total	* Starting A* Search with Custom Heuristic * A* search succeeded found plan as follows: 00: (load_box_in_warehouse robot1 box2 valve warehouse) [0] 01: (pick_box_from_loc robot1 box2 warehouse) [0] 02: (move_robot robot1 warehouse location1) [0] 03: (move_robot robot1 location1 location2) [0] 04: (enter_workstation robot1 location2 ws3) [0] 05: (drop_box_in_ws robot1 box2 ws3) [0] 06: (unload_box_in_ws robot1 box2 ws3 valve) [0] 07: (exit_workstation robot1 location2 ws3) [0] 08: (move_robot robot1 location2 location1) [0] 09: (move_robot robot1 location1 warehouse) [0] 10: (load_box_in_warehouse robot1 box3 bolt warehouse) [0] 11: (pick_box_from_loc robot1 box3 warehouse) [0] 12: (move_robot robot1 warehouse location1) [0] 13: (enter_workstation robot1 location1 ws1) [0] 14: (drop_box_in_ws robot1 box3 ws1) [0] 15: (unload_box_in_ws robot1 box3 ws1 bolt) [0] 16: (exit_workstation robot1 location1 ws1) [0] 17: (move_robot robot1 location1 warehouse) [0] 18: (load_box_in_warehouse robot1 box1 bolt warehouse) [0] 19: (pick_box_from_loc robot1 box1 warehouse) [0] 20: (move_robot robot1 warehouse location1) [0] 21: (move_robot robot1 location1 location2) [0] 22: (enter_workstation robot1 location2 ws2) [0] 23: (drop_box_in_ws robot1 box1 ws2) [0] 24: (unload_box_in_ws robot1 box1 ws2 bolt) [0] 25: (pick_box_from_ws robot1 box1 ws2) [0] 26: (exit_workstation robot1 location2 ws2) [0] 27: (move_robot robot1 location2 location1) [0] 28: (move_robot robot1 location1 warehouse) [0] 29: (drop_box_in_loc robot1 box1 warehouse) [0] 30: (load_box_in_warehouse robot1 box1 valve warehouse) [0] 31: (pick_box_from_loc robot1 box1 warehouse) [0] 32: (move_robot robot1 warehouse location1) [0] 33: (enter_workstation robot1 location1 ws1) [0] 34: (drop_box_in_ws robot1 box1 ws1) [0] 35: (unload_box_in_ws robot1 box1 ws1 valve) [0] 36: (pick_box_from_ws robot1 box1 ws1) [0] 37: (exit_workstation robot1 location1 ws1) [0] 38: (move_robot robot1 location1 warehouse) [0] 39: (drop_box_in_loc robot1 box1 warehouse) [0] 40: (load_box_in_warehouse robot1 box1 bolt warehouse) [0] 41: (pick_box_from_loc robot1 box1 warehouse) [0] 42: (move_robot robot1 warehouse location1) [0] 43: (move_robot robot1 location1 location2) [0] 44: (enter_workstation robot1 location2 ws3) [0] 45: (drop_box_in_ws robot1 box1 ws3) [0] 46: (unload_box_in_ws robot1 box1 ws3 bolt) [0] time spent: 0,02 seconds parsing 0,03 seconds encoding 443,43 seconds searching 443,47 seconds total time memory used: 0,51 MBytes for problem representation 1943,50 MBytes for searching 1944,01 MBytes total

Tabella riassuntiva con i risultati ottenuti:

Algoritmo	Num. Azioni	Tempo Totale	Spazio Totale
FF	48	0.56 s	0.52 MB
A* + FF	46	36.11 s	488.88 MB
A* + AS	46	23.04 s	390.76 MB
A* + Custom	46	443.47 s	1944.01 MB

2.5.2 Istanza 2

FF

```
* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD heuristic
* ENFORCED_HILL_CLIMBING search succeeded

found plan as follows:

00: ( load_box_in_warehouse robot box1 bolt warehouse carrier) [0]
01: (    pick_box_from_loc robot box1 warehouse carrier slot1) [0]
02: (load_box_in_warehouse robot box2 valve warehouse carrier) [0]
03: (    pick_box_from_loc robot box2 warehouse carrier slot2) [0]
04: (            move_robot robot warehouse location1 carrier) [0]
05: (            move_robot robot location1 location2 carrier) [0]
06: (  enter_workstation robot location2 workstation3 carrier) [0]
07: (    drop_box_in_ws robot box2 workstation3 carrier slot1) [0]
08: (  unload_box_in_ws robot box2 workstation3 valve carrier) [0]
09: (    drop_box_in_ws robot box1 workstation3 carrier slot2) [0]
10: (   unload_box_in_ws robot box1 workstation3 bolt carrier) [0]
11: (   exit_workstation robot location2 workstation3 carrier) [0]
12: (            move_robot robot location2 location1 carrier) [0]
13: (  move_robot_warehouse robot location1 warehouse carrier) [0]
14: ( load_box_in_warehouse robot box3 bolt warehouse carrier) [0]
15: (    pick_box_from_loc robot box3 warehouse carrier slot1) [0]
16: (            move_robot robot warehouse location1 carrier) [0]
17: (            move_robot robot location1 location2 carrier) [0]
18: (  enter_workstation robot location2 workstation2 carrier) [0]
19: (    drop_box_in_ws robot box3 workstation2 carrier slot1) [0]
20: (   unload_box_in_ws robot box3 workstation2 bolt carrier) [0]

time spent:       0,02 seconds parsing
                  0,03 seconds encoding
                  1,91 seconds searching
                  1,96 seconds total time

memory used:      0,76 MBytes for problem representation
                  0,00 MBytes for searching
                  0,76 MBytes total
```

A* con euristica FF

```
* Starting A* search with heuristic: FAST_FORWARD
* Starting A* search
* A* search succeeded

found plan as follows:

00: (load_box_in_warehouse robot box3 valve warehouse carrier) [0]
01: (    pick_box_from_loc robot box3 warehouse carrier slot2) [0]
02: ( load_box_in_warehouse robot box1 bolt warehouse carrier) [0]
03: (    pick_box_from_loc robot box1 warehouse carrier slot3) [0]
04: ( load_box_in_warehouse robot box2 bolt warehouse carrier) [0]
05: (    pick_box_from_loc robot box2 warehouse carrier slot1) [0]
06: (            move_robot robot warehouse location1 carrier) [0]
07: (            move_robot robot location1 location2 carrier) [0]
08: (  enter_workstation robot location2 workstation2 carrier) [0]
09: (    drop_box_in_ws robot box2 workstation2 carrier slot1) [0]
10: (   unload_box_in_ws robot box2 workstation2 bolt carrier) [0]
11: (   exit_workstation robot location2 workstation2 carrier) [0]
12: (  enter_workstation robot location2 workstation3 carrier) [0]
13: (    drop_box_in_ws robot box1 workstation3 carrier slot2) [0]
14: (    drop_box_in_ws robot box3 workstation3 carrier slot3) [0]
15: (   unload_box_in_ws robot box1 workstation3 bolt carrier) [0]
16: (  unload_box_in_ws robot box3 workstation3 valve carrier) [0]

time spent:       0,02 seconds parsing
                  0,03 seconds encoding
                  0,23 seconds searching
                  0,28 seconds total time

memory used:      0,76 MBytes for problem representation
                  1,80 MBytes for searching
                  2,56 MBytes total
```

A* con euristica AS	A* con euristica Custom
* Starting A* search with heuristic: AJUSTED_SUM * Starting A* search * A* search succeeded found plan as follows: 00: (load_box_in_warehouse robot box1 bolt warehouse carrier) [0] 01: (load_box_in_warehouse robot box3 bolt warehouse carrier) [0] 02: (load_box_in_warehouse robot box2 valve warehouse carrier) [0] 03: (pick_box_from_loc robot box1 warehouse carrier slot2) [0] 04: (pick_box_from_loc robot box3 warehouse carrier slot3) [0] 05: (pick_box_from_loc robot box2 warehouse carrier slot1) [0] 06: (move_robot robot warehouse location1 carrier) [0] 07: (move_robot robot location1 location2 carrier) [0] 08: (enter_workstation robot location2 workstation3 carrier) [0] 09: (drop_box_in_ws robot box1 workstation3 carrier slot2) [0] 10: (drop_box_in_ws robot box2 workstation3 carrier slot3) [0] 11: (unload_box_in_ws robot box2 workstation3 valve carrier) [0] 12: (unload_box_in_ws robot box1 workstation3 bolt carrier) [0] 13: (exit_workstation robot location2 workstation3 carrier) [0] 14: (enter_workstation robot location2 workstation2 carrier) [0] 15: (drop_box_in_ws robot box3 workstation2 carrier slot1) [0] 16: (unload_box_in_ws robot box3 workstation2 bolt carrier) [0] time spent: 0,02 seconds parsing 0,03 seconds encoding 1,65 seconds searching 1,70 seconds total time memory used: 0,76 MBytes for problem representation 25,24 MBytes for searching 25,99 MBytes total	* Starting A* Search with Custom Heuristic * A* search succeeded found plan as follows: 00: (load_box_in_warehouse robot box2 bolt warehouse carrier) [0] 01: (load_box_in_warehouse robot box1 bolt warehouse carrier) [0] 02: (load_box_in_warehouse robot box3 valve warehouse carrier) [0] 03: (pick_box_from_loc robot box3 warehouse carrier slot3) [0] 04: (pick_box_from_loc robot box2 warehouse carrier slot2) [0] 05: (pick_box_from_loc robot box1 warehouse carrier slot1) [0] 06: (move_robot robot warehouse location1 carrier) [0] 07: (move_robot robot location1 location2 carrier) [0] 08: (enter_workstation robot location2 workstation3 carrier) [0] 09: (drop_box_in_ws robot box3 workstation3 carrier slot3) [0] 10: (unload_box_in_ws robot box3 workstation3 valve carrier) [0] 11: (drop_box_in_ws robot box1 workstation3 carrier slot3) [0] 12: (unload_box_in_ws robot box1 workstation3 bolt carrier) [0] 13: (exit_workstation robot location2 workstation3 carrier) [0] 14: (enter_workstation robot location2 workstation2 carrier) [0] 15: (drop_box_in_ws robot box2 workstation2 carrier slot3) [0] 16: (unload_box_in_ws robot box2 workstation2 bolt carrier) [0] time spent: 0,02 seconds parsing 0,03 seconds encoding 513,89 seconds searching 513,93 seconds total time memory used: 0,76 MBytes for problem representation 1685,25 MBytes for searching 1686,01 MBytes total

Tabella riassuntiva con i risultati ottenuti:

Algoritmo	Num. Azioni	Tempo Totale	Spazio Totale
FF	48	0.56 s	0.52 MB
A* + FF	46	36.11 s	488.88 MB
A* + AS	46	23.04 s	390.76 MB
A* + Custom	46	443.47 s	1944.01 MB

3 Task 3

Il *Task3* si suddivide in due punti:

1. **Temporal Planning**: convertire il dominio definito nella sezione 2.2 al fine di utilizzare le "*durative action*", ovvero azioni che hanno una durata temporale.
2. **Robotics Planning**: implementare il problema utilizzando *PlanSys2* utilizzando le "*fake action*".

3.1 Temporal Planning

3.1.1 Dominio

In questa sezione, come detto in precedenza, viene effettuata la conversione del dominio definito nella sezione 2.2.1, affinché si possa generare una sequenza temporale di azioni caratterizzate da una durata. Un problema di pianificazione temporale riguarda l'identificazione di una sequenza di azioni che, oltre a rispettare i requisiti di eseguibilità causale (come avviene nella pianificazione classica), soddisfino anche una serie di vincoli temporali relativi alla durata delle azioni, lungo una linea temporale senza limiti. Un'azione durativa rappresenta un tipo di azione che necessita di un determinato intervallo di tempo per essere completata, il quale può essere definito come un valore fisso o variabile. Questo cambiamento semantico è stato introdotto per rappresentare il fatto che un'azione durativa può avere condizioni che devono essere vere non solo all'inizio dell'azione, ma anche alla fine o per tutta la sua durata, in supporto di ciò ci vengono in aiuto dei costrutti:

- *At start*: seguito da un predicato specifica che la condizione espressa deve essere valida prima dell'inizio dell'azione.
- *Over all*: specifica che la condizione espressa deve essere valida per tutta la durata dell'azione, dall'inizio fino al suo completamento.
- *At end*: specifica che la condizione espressa deve essere valida alla conclusione dell'azione.

Si mostra di seguito il dominio aggiornato con l'aggiunta delle *durative action*:

```
(define (domain manufacturing_temporal)

    (:requirements :strips :typing :equality :adl :durative-actions :universal-
preconditions)

    ;; Tipologie degli oggetti gestiti
    (:types
```

```
        location box content robot workstation carrier slot
    )


    ;; Predicati
    (:predicates
        (located_at ?entity - (either robot workstation box content carrier) ?loc -
location)
        (is_empty ?box - box)
        (contains ?box - box ?item - content)
        (inside_ws ?entity - (either robot box content) ?ws - workstation)
        (carrying ?carrier - carrier ?box - box)
        (linked ?loc1 ?loc2 - location)
        (is_warehouse ?loc - location)
        (joined ?robot - robot ?carrier - carrier)
        (handle ?carrier - carrier ?slot - slot)
        (available ?slot - slot)
    )


    ;; Durative Actions

    ; Spostare il robot tra due location adiacenti
    (:durative-action move_robot
        :parameters (?r - robot ?from ?to - location ?c - carrier)
        :duration (= ?duration 3)
        :condition (and (at start (located_at ?r ?from))
                        (at start (linked ?from ?to))
                        (over all (joined ?r ?c)))
        :effect (and (at start (not (located_at ?r ?from)))
                     (at end (located_at ?r ?to)))
    )

    ; Spostare il robot verso il magazzino
    (:durative-action move_robot_warehouse
        :parameters (?r - robot ?from ?to - location ?c - carrier)
        :duration (= ?duration 3)
        :condition (and (at start (located_at ?r ?from))
                        (at start (linked ?from ?to))
                        (at start (is_warehouse ?to))
                        (over all (joined ?r ?c))
                        (at start (forall (?slot - slot) (and (handle ?c ?slot)
(available ?slot)))))
        :effect (and (at start (not (located_at ?r ?from)))
                     (at end (located_at ?r ?to)))
    )
```

```
; Il robot entra in una workstation
(:durative-action enter_workstation
    :parameters (?r - robot ?loc - location ?ws - workstation ?c - carrier)
    :duration (= ?duration 0)
    :condition (and (at start (located_at ?ws ?loc))
                    (at start (located_at ?r ?loc))
                    (over all (joined ?r ?c)))
    :effect (and (at start (not (located_at ?r ?loc)))
                 (at end (inside_ws ?r ?ws)))
)

; Il robot esce dalla workstation
(:durative-action exit_workstation
    :parameters (?r - robot ?loc - location ?ws - workstation ?c - carrier)
    :duration (= ?duration 0)
    :condition (and (at start (inside_ws ?r ?ws))
                    (over all (joined ?r ?c)))
    :effect (and (at start (not (inside_ws ?r ?ws)))
                 (at end (located_at ?r ?loc)))
)

; Il robot lascia una scatola in una workstation
(:durative-action drop_box_in_ws
    :parameters (?r - robot ?box - box ?ws - workstation ?c - carrier ?slot - slot)
    :duration (= ?duration 2)
    :condition (and (at start (inside_ws ?r ?ws))
                    (at start (carrying ?c ?box))
                    (at start (joined ?r ?c))
                    (at start (handle ?c ?slot)))
    :effect (and (at end (not (carrying ?c ?box)))
                 (at end (available ?slot))
                 (at end (inside_ws ?box ?ws)))
)

; Il robot lascia una scatola in una location
(:durative-action drop_box_in_loc
    :parameters (?r - robot ?box - box ?loc - location ?c - carrier ?slot - slot)
    :duration (= ?duration 2)
    :condition (and (at start (located_at ?r ?loc))
                    (at start (carrying ?c ?box))
                    (at start (joined ?r ?c))
                    (at start (handle ?c ?slot)))
    :effect (and (at end (not (carrying ?c ?box)))
                 (at end (available ?slot))
```

```
                    (at end (located_at ?box ?loc)))
    )

    ; Il robot raccoglie una scatola da una workstation
    (:durative-action pick_box_from_ws
        :parameters (?r - robot ?box - box ?ws - workstation ?c - carrier ?slot - slot)
        :duration (= ?duration 2)
        :condition (and (at start (inside_ws ?box ?ws))
                        (at start (inside_ws ?r ?ws))
                        (at start (joined ?r ?c))
                        (at start (handle ?c ?slot))
                        (at start (available ?slot)))
        :effect (and (at end (carrying ?c ?box))
                     (at end (not (available ?slot)))
                     (at end (not (inside_ws ?box ?ws))))
    )

    ; Il robot raccoglie una scatola da una location
    (:durative-action pick_box_from_loc
        :parameters (?r - robot ?box - box ?loc - location ?c - carrier ?slot - slot)
        :duration (= ?duration 2)
        :condition (and (at start (located_at ?box ?loc))
                        (at start (located_at ?r ?loc))
                        (at start (joined ?r ?c))
                        (at start (handle ?c ?slot))
                        (at start (available ?slot)))
        :effect (and (at end (carrying ?c ?box))
                     (at end (not (available ?slot)))
                     (at end (not (located_at ?box ?loc))))
    )

    ; Il robot svuota una scatola in una workstation
    (:durative-action unload_box_in_ws
        :parameters (?r - robot ?box - box ?ws - workstation ?content - content ?c -
carrier)
        :duration (= ?duration 3.5)
        :condition (and (at start (inside_ws ?r ?ws))
                        (at start (inside_ws ?box ?ws))
                        (at start (contains ?box ?content))
                        (over all (joined ?r ?c)))
        :effect (and (at end (not (contains ?box ?content)))
                     (at end (is_empty ?box))
                     (at end (inside_ws ?content ?ws)))
    )
```

```
    ; Il robot riempie una scatola in una workstation
    (:durative-action load_box_in_ws
        :parameters (?r - robot ?box - box ?ws - workstation ?content - content ?c -
carrier)
        :duration (= ?duration 3.5)
        :condition (and (at start (inside_ws ?r ?ws))
                        (at start (inside_ws ?box ?ws))
                        (at start (inside_ws ?content ?ws))
                        (at start (is_empty ?box))
                        (over all (joined ?r ?c)))
        :effect (and (at end (not (inside_ws ?content ?ws)))
                     (at end (contains ?box ?content))
                     (at end (not (is_empty ?box))))
    )

    ; Il robot riempie una scatola in una location
    (:durative-action load_box_in_loc
        :parameters (?r - robot ?box - box ?content - content ?loc - location ?c -
carrier)
        :duration (= ?duration 3.5)
        :condition (and (at start (located_at ?r ?loc))
                        (at start (located_at ?box ?loc))
                        (at start (located_at ?content ?loc))
                        (at start (is_empty ?box))
                        (over all (joined ?r ?c)))
        :effect (and (at end (not (located_at ?content ?loc)))
                     (at end (contains ?box ?content))
                     (at end (not (is_empty ?box))))
    )

    ; Il robot svuota una scatola in una location
    (:durative-action unload_box_in_loc
        :parameters (?r - robot ?box - box ?content - content ?loc - location ?c -
carrier)
        :duration (= ?duration 3.5)
        :condition (and (at start (located_at ?r ?loc))
                        (at start (located_at ?box ?loc))
                        (at start (contains ?box ?content))
                        (over all (joined ?r ?c)))
        :effect (and (at end (not (contains ?box ?content)))
                     (at end (is_empty ?box))
                     (at end (located_at ?content ?loc)))
    )

    ; Il robot riempie una scatola nel magazzino
```

```
    (:durative-action load_box_in_warehouse
        :parameters (?r - robot ?box - box ?content - content ?loc - location ?c -
carrier)
        :duration (= ?duration 3.5)
        :condition (and (at start (is_warehouse ?loc))
                        (at start (located_at ?r ?loc))
                        (at start (located_at ?box ?loc))
                        (at start (located_at ?content ?loc))
                        (at start (is_empty ?box))
                        (over all (joined ?r ?c)))
        :effect (and (at end (contains ?box ?content))
                     (at end (not (is_empty ?box))))
    )
)
```

3.1.2 Risultati

Il problema utilizzato per la risoluzione è quello precedentemente descritto nella sezione 2.2.2.

Per la risoluzione è stato utilizzato il framework *planutils*, scegliendo come planner risolutore **LPG-td**, in quanto supporta le durative action ed è stato in grado di risolvere il problema in tempi brevi.

È stata eseguita la seguente linea di comando:

```
/home/aiguy/.planutils/packages/lpg-td/bin/lpg-td -o domain.pddl -f problem.pddl -v off
-noout -quality
```

Producendo il seguente plan:

```
Plan computed:
   Time: (ACTION) [action Duration; action Cost]
 0.0003: (LOAD_BOX_IN_WAREHOUSE ROBOT BOX1 VALVE WAREHOUSE CARRIER) [D:3.5000; C:1.0000]
 0.0005: (PICK_BOX_FROM_LOC ROBOT BOX1 WAREHOUSE CARRIER SLOT2) [D:2.0000; C:1.0000]
 0.0008: (LOAD_BOX_IN_WAREHOUSE ROBOT BOX3 BOLT WAREHOUSE CARRIER) [D:3.5000; C:1.0000]
 0.0010: (PICK_BOX_FROM_LOC ROBOT BOX3 WAREHOUSE CARRIER SLOT3) [D:2.0000; C:1.0000]
 0.0012: (LOAD_BOX_IN_LOC ROBOT BOX2 BOLT WAREHOUSE CARRIER) [D:3.5000; C:1.0000]
 0.0015: (PICK_BOX_FROM_LOC ROBOT BOX2 WAREHOUSE CARRIER SLOT1) [D:2.0000; C:1.0000]
 0.0018: (MOVE_ROBOT ROBOT WAREHOUSE LOCATION1 CARRIER) [D:3.0000; C:1.0000]
 3.0020: (MOVE_ROBOT ROBOT LOCATION1 LOCATION2 CARRIER) [D:3.0000; C:1.0000]
 6.0023: (ENTER_WORKSTATION ROBOT LOCATION2 WORKSTATION3 CARRIER) [D:0.0000; C:1.0000]
 6.0025: (DROP_BOX_IN_WS ROBOT BOX1 WORKSTATION3 CARRIER SLOT3) [D:2.0000; C:1.0000]
 8.0028: (UNLOAD_BOX_IN_WS ROBOT BOX1 WORKSTATION3 VALVE CARRIER) [D:3.5000; C:1.0000]
 6.0030: (DROP_BOX_IN_WS ROBOT BOX3 WORKSTATION3 CARRIER SLOT3) [D:2.0000; C:1.0000]
 8.0033: (UNLOAD_BOX_IN_WS ROBOT BOX3 WORKSTATION3 BOLT CARRIER) [D:3.5000; C:1.0000]
 8.0035: (EXIT_WORKSTATION ROBOT LOCATION2 WORKSTATION3 CARRIER) [D:0.0000; C:1.0000]
 8.0037: (ENTER_WORKSTATION ROBOT LOCATION2 WORKSTATION2 CARRIER) [D:0.0000; C:1.0000]
 8.0040: (DROP_BOX_IN_WS ROBOT BOX2 WORKSTATION2 CARRIER SLOT2) [D:2.0000; C:1.0000]
 10.0042: (UNLOAD_BOX_IN_WS ROBOT BOX2 WORKSTATION2 BOLT CARRIER) [D:3.5000; C:1.0000]
```

```
Solution found:
Total time:      5.26
Search time:     0.26
Actions:         17
Execution cost:  17.00
Duration:        456.500
Plan quality:    17.000
     Plan file:       plan_problem.pddl_1.SOL
```

Il piano generato sembra quindi ragionevolmente buono rispetto agli obiettivi e ai vincoli imposti dal problema.

3.2 Robotics Planning

L'ultima sezione è dedicata alla pianificazione robotica, un ambito che si occupa della progettazione e dello sviluppo di algoritmi e strategie per consentire ai robot autonomi di pianificare e programmare le proprie azioni al fine di raggiungere obiettivi specifici in ambienti dinamici e incerti. Per pianificare le proprie azioni, un robot deve disporre di una rappresentazione accurata dell'ambiente in cui opera e deve essere in grado di elaborare un piano che lo guidi verso il raggiungimento degli obiettivi prefissati.

Per affrontare il secondo punto del terzo task richiesto, è necessario implementare il problema descritto nella sezione precedente utilizzando il *ROS2 Planning System* (Plansys2), una piattaforma open-source basata su ROS (Robot Operating System 2). Plansys2 offre strumenti per la pianificazione, l'esecuzione e la gestione delle attività dei robot autonomi, con supporto per la pianificazione temporale.

All'interno di Plansys2 si trovano le "*fake actions*", ovvero azioni simulate o rappresentate in modo da permettere al sistema di pianificazione di gestire situazioni specifiche o modellare comportamenti desiderati che non corrispondono necessariamente ai comportamenti reali del robot o dell'agente. Queste azioni vengono utilizzate per rappresentare situazioni particolari, come il fatto che il robot "attenda" in un determinato stato o posizione per un certo periodo di tempo. Le *fake actions* sono utili per gestire le tempistiche, includendo pause o ritardi.

Le *fake actions* sono state definite nel linguaggio C++, ad ogni fake action corrisponde una classe, di seguito viene mostrato un esempio:

```cpp
#include <memory>
#include <algorithm>
#include<vector>
#include<string>

#include "plansys2_executor/ActionExecutorClient.hpp"

#include "rclcpp/rclcpp.hpp"
#include "rclcpp_action/rclcpp_action.hpp"

using namespace std::chrono_literals;

class MoveRobot : public plansys2::ActionExecutorClient
{
public:
  MoveRobot()
  : plansys2::ActionExecutorClient("moverobot", 250ms)
  {
    progress_ = 0.0;
  }

private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.2;
      send_feedback(progress_, "Robot "+arguments [0]+ " moving from "+
            arguments [1]+ " to "+ arguments [2] + " carring a "+
            arguments [3]);
    } else {
      finish(true, 1.0, "Robot "+arguments [0]+ " moving from "+
            arguments [1]+ " to "+ arguments [2] + " carring a "+
            arguments [3]);

      progress_ = 0.0;
      std::cout << std::endl;
    }

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Robot "+arguments [0]+ " moving from "+
            arguments [1]+ " to "+ arguments [2] + " carring a "+
            arguments [3]+" . . . [ " << std::min(100.0, progress_ * 100.0) << "% ]  " <<
            std::flush;
  }

  float progress_;
};

int main(int argc, char ** argv)
{
  rclcpp::init(argc, argv);
```

```
  auto node = std::make_shared<MoveRobot>();

  node->set_parameter(rclcpp::Parameter("action_name", "MOVEROBOT"));
  node->trigger_transition(lifecycle_msgs::msg::Transition::TRANSITION_CONFIGURE);

  rclcpp::spin(node->get_node_base_interface());

  rclcpp::shutdown();

  return 0;
}
```

Per procedere nella risoluzione del task è stata necessaria la creazione di un file di launch in linguaggio Python, che viene riportato di seguito:

```
import os

from ament_index_python.packages import get_package_share_directory

from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import LaunchConfiguration
from launch_ros.actions import Node


def generate_launch_description():
    # Get the launch directory
    example_dir = get_package_share_directory('manufactoring_service_robots')
    namespace = LaunchConfiguration('namespace')

    declare_namespace_cmd = DeclareLaunchArgument(
        'namespace',
        default_value='',
        description='Namespace')

    plansys2_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(os.path.join(
            get_package_share_directory('plansys2_bringup'),
            'launch',
            'plansys2_bringup_launch_monolithic.py')),
        launch_arguments={
          'model_file': example_dir + '/pddl/domain.pddl',
          'namespace': namespace
          }.items())
```

```
# Specify the actions
unload_box_in_loc_cmd = Node(
    package='manufactoring_service_robots',
    executable='unload_box_in_loc_action_node',
    name='unload_box_in_loc_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

unload_box_in_ws_cmd = Node(
    package='manufactoring_service_robots',
    executable='unload_box_in_ws_action_node',
    name='unload_box_in_ws_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

enter_workstation_cmd = Node(
    package='manufactoring_service_robots',
    executable='enter_workstation_action_node',
    name='enter_workstation_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])   # Create the launch description and populate

exit_workstation_cmd = Node(
    package='manufactoring_service_robots',
    executable='exit_workstation_action_node',
    name='exit_workstation_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

load_box_in_loc_cmd = Node(
    package='manufactoring_service_robots',
    executable='load_box_in_loc_action_node',
    name='load_box_in_loc_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

load_box_in_warehouse_cmd = Node(
    package='manufactoring_service_robots',
    executable='load_box_in_warehouse_action_node',
    name='load_box_in_warehouse_action_node',
```

```
        namespace=namespace,
        output='screen',
        parameters=[])

    load_box_in_ws_cmd = Node(
        package='manufactoring_service_robots',
        executable='load_box_in_ws_action_node',
        name='load_box_in_ws_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    pick_box_from_loc_cmd = Node(
        package='manufactoring_service_robots',
        executable='pick_box_from_loc_action_node',
        name='pick_box_from_loc_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    pick_box_from_ws_cmd = Node(
        package='manufactoring_service_robots',
        executable='pick_box_from_ws_action_node',
        name='pick_box_from_ws_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    move_robot_cmd = Node(
        package='manufactoring_service_robots',
        executable='move_robot_action_node',
        name='move_robot_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])


    move_robot_warehouse_cmd = Node(
        package='manufactoring_service_robots',
        executable='move_robot_warehouse_action_node',
        name='move_robot_warehouse_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])
```

```
drop_box_in_loc_cmd = Node(
    package='manufactoring_service_robots',
    executable='drop_box_in_loc_action_node',
    name='drop_box_in_loc_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])


drop_box_in_ws_cmd = Node(
    package='manufactoring_service_robots',
    executable='drop_box_in_ws_action_node',
    name='drop_box_in_ws_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

ld = LaunchDescription()

ld.add_action(declare_namespace_cmd)

# Declare the launch options
ld.add_action(plansys2_cmd)

ld.add_action(unload_box_in_loc_cmd)
ld.add_action(unload_box_in_ws_cmd)
ld.add_action(enter_workstation_cmd)
ld.add_action(exit_workstation_cmd)
ld.add_action(load_box_in_loc_cmd)
ld.add_action(load_box_in_warehouse_cmd)
ld.add_action(load_box_in_ws_cmd)
ld.add_action(pick_box_from_loc_cmd)
ld.add_action(pick_box_from_ws_cmd)
ld.add_action(move_robot_cmd)
ld.add_action(move_robot_warehouse_cmd)
ld.add_action(drop_box_in_loc_cmd)
ld.add_action(drop_box_in_ws_cmd)

return ld
```

Fatto ciò, si è proceduto a descrivere l'istanza del problema da risolvere all'interno del file *command*, facendo riferimento al problem descritto precedentemente nella sezione 2.2.2, riportata di seguito:

```
set instance ROBOT robot

set instance CARRIER carrier

set instance WAREHOUSE location
set instance LOCATION1 location
set instance LOCATION2 location

set instance WORKSTATION1 workstation
set instance WORKSTATION2 workstation
set instance WORKSTATION3 workstation

set instance BOX1 box
set instance BOX2 box
set instance BOX3 box

set instance BOLT content
set instance VALVE content

set instance SLOT1 slot
set instance SLOT2 slot
set instance SLOT3 slot

set predicate (located_at ROBOT WAREHOUSE)

set predicate (joined ROBOT CARRIER)

set predicate (is_warehouse WAREHOUSE)

set predicate (located_at BOX1 WAREHOUSE)
set predicate (located_at BOX2 WAREHOUSE)
set predicate (located_at BOX3 WAREHOUSE)

set predicate (located_at BOLT WAREHOUSE)
set predicate (located_at VALVE WAREHOUSE)

set predicate (is_empty BOX1)
set predicate (is_empty BOX2)
set predicate (is_empty BOX3)

set predicate (located_at WORKSTATION1 LOCATION1)
```

```
set predicate (located_at WORKSTATION2 LOCATION2)
set predicate (located_at WORKSTATION3 LOCATION2)

set predicate (linked WAREHOUSE LOCATION1)
set predicate (linked LOCATION1 LOCATION2)
set predicate (linked LOCATION1 WAREHOUSE)
set predicate (linked LOCATION2 LOCATION1)

set predicate (handle CARRIER SLOT1)
set predicate (handle CARRIER SLOT2)
set predicate (handle CARRIER SLOT3)

set predicate (available SLOT1)
set predicate (available SLOT2)
set predicate (available SLOT3)
```

L'ultima operazione preliminare è stata quella della configurazione del file *CMakeList*, anch'essa riportata di seguito:

```
cmake_minimum_required(VERSION 3.5)
project(manufactoring_service_robots)


find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(rclcpp_action REQUIRED)
find_package(plansys2_msgs REQUIRED)
find_package(plansys2_executor REQUIRED)

set(CMAKE_CXX_STANDARD 17)

set(dependencies
    rclcpp
    rclcpp_action
    plansys2_msgs
    plansys2_executor
)

add_executable(unload_box_in_loc_action_node src/unload_box_in_loc_action_node.cpp)
ament_target_dependencies(unload_box_in_loc_action_node ${dependencies})

add_executable(unload_box_in_ws_action_node src/unload_box_in_ws_action_node.cpp)
ament_target_dependencies(unload_box_in_ws_action_node ${dependencies})
```

```
add_executable(enter_workstation_action_node src/enter_workstation_action_node.cpp)
ament_target_dependencies(enter_workstation_action_node ${dependencies})

add_executable(exit_workstation_action_node src/exit_workstation_action_node.cpp)
ament_target_dependencies(exit_workstation_action_node ${dependencies})

add_executable(load_box_in_loc_action_node src/load_box_in_loc_action_node.cpp)
ament_target_dependencies(load_box_in_loc_action_node ${dependencies})

add_executable(load_box_in_warehouse_action_node
src/load_box_in_warehouse_action_node.cpp)
ament_target_dependencies(load_box_in_warehouse_action_node ${dependencies})

add_executable(load_box_in_ws_action_node src/load_box_in_ws_action_node.cpp)
ament_target_dependencies(load_box_in_ws_action_node ${dependencies})

add_executable(pick_box_from_loc_action_node src/pick_box_from_loc_action_node.cpp)
ament_target_dependencies(pick_box_from_loc_action_node ${dependencies})

add_executable(pick_box_from_ws_action_node src/pick_box_from_ws_action_node.cpp)
ament_target_dependencies(pick_box_from_ws_action_node ${dependencies})

add_executable(move_robot_action_node src/move_robot_action_node.cpp)
ament_target_dependencies(move_robot_action_node ${dependencies})

add_executable(move_robot_warehouse_action_node
src/move_robot_warehouse_action_node.cpp)
ament_target_dependencies(move_robot_warehouse_action_node ${dependencies})

add_executable(drop_box_in_loc_action_node src/unload_box_in_loc_action_node.cpp)
ament_target_dependencies(unload_box_in_loc_action_node ${dependencies})

add_executable(drop_box_in_ws_action_node src/unload_box_in_ws_action_node.cpp)
ament_target_dependencies(unload_box_in_ws_action_node ${dependencies})


install(DIRECTORY launch pddl DESTINATION share/${PROJECT_NAME})

install(TARGETS
  unload_box_in_loc_action_node
  unload_box_in_ws_action_node
  enter_workstation_action_node
  exit_workstation_action_node
  load_box_in_loc_action_node
```

```
  load_box_in_warehouse_action_node
  load_box_in_ws_action_node
  pick_box_from_loc_action_node
  pick_box_from_ws_action_node
  move_robot_action_node
  move_robot_warehouse_action_node
  drop_box_in_loc_action_node
  drop_box_in_ws_action_node
  ARCHIVE DESTINATION lib
  LIBRARY DESTINATION lib
  RUNTIME DESTINATION lib/${PROJECT_NAME}
)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  ament_lint_auto_find_test_dependencies()

  find_package(ament_cmake_gtest REQUIRED)
endif()

ament_export_dependencies(${dependencies})

ament_package()
```

A questo punto si può procedere a lanciare il sistema realizzato, utilizzando il plan generato precedentemente, mostrato nella sezione 3.1.2, si ottiene il seguente risultato:

```
[load_box_in_warehouse_action_node-3] box2 in warehouse with bolt . . . [100%]

[move_robot_action_node-9] ROBOT moving from warehouse to loc4 . . . [100%]

[pick_box_from_loc_action_node-5] ROBOT is picking up box2 in warehouse . . . [100%]

[drop_box_in_ws_action_node-12] ROBOT is delivering box2 to ws1 . . . [100%]

[unload_box_in_ws_action_node-6] ROBOT is emptying box2 in ws1, the box contained bolt . . . [100%]

[move_robot_action_node-10] ROBOT moving from loc1 to warehouse . . . [100%]

[load_box_in_warehouse_action_node-3] box1 in warehouse with valve . . . [100%]

[move_robot_action_node-9] ROBOT moving from warehouse to loc2 . . . [100%]

[pick_box_from_loc_action_node-5] ROBOT is picking up box1 in warehouse . . . [100%]

[drop_box_in_ws_action_node-12] ROBOT is delivering box1 to ws3 . . . [100%]
```

```
[unload_box_in_ws_action_node-6] ROBOT is emptying box1 in ws3, the box contained valve . . . [100%]

[move_robot_action_node-10] ROBOT moving from loc3 to warehouse . . . [100%]

[load_box_in_warehouse_action_node-3] box3 in warehouse with screw . . . [100%]

[move_robot_action_node-9] ROBOT moving from warehouse to loc3 . . . [100%]

[pick_box_from_loc_action_node-5] ROBOT is picking up box3 in warehouse . . . [100%]

[drop_box_in_ws_action_node-12] ROBOT is delivering box3 to ws3 . . . [100%]

[unload_box_in_ws_action_node-6] ROBOT is emptying box3 in ws3, the box contained screw . . . [100%]

[move_robot_action_node-10] ROBOT moving from loc3 to loc2 . . . [100%]

[drop_box_in_ws_action_node-12] ROBOT is delivering box3 to ws2 . . . [100%]

[unload_box_in_ws_action_node-6] ROBOT is emptying box3 in ws2, the box contained screw . . . [100%]

[plansys2_node-1] [INFO] [executor]: Plan Succeeded
```

Per lanciare il sistema e quindi ottenere tali risultati è necessario eseguire in ordine i seguenti passaggi, utilizzando due terminali:

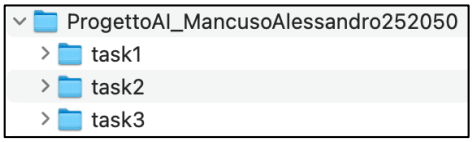
- *Primo Terminale:*
  - `colcon build --symlink-install`
  - `source install/setup.bash`
  - `ros2 launch manufactoring_service_robots plansys2_msl.py`

A questo punto si apre un secondo terminale all'interno di `3.2/src/manufactoring_service_robots`

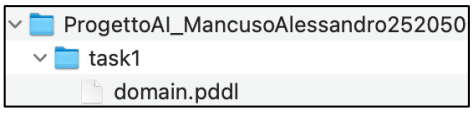
- *Secondo Terminale:*
  - `colcon build --symlink-install`
  - `source ./launch/commands`
  - `run plan-file <planPath>`

4 Deliverable

L'archivio è organizzato in 3 cartelle, una per ciascun task:

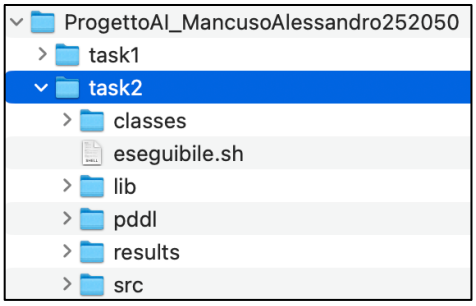


Cartella *task1*: all'interno della cartella è presente il file *domain.pddl* descritto nella sezione 1.1

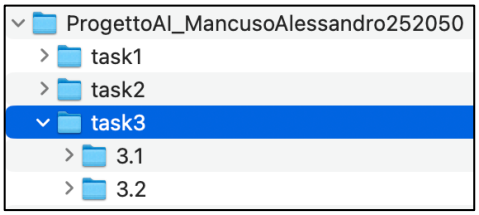


Cartella *task2*: all'interno della cartella è presente uno script *eseguibile.sh* che permette di avviare facilmente la risoluzione dei problemi, inoltre sono presenti le seguenti sottocartelle:

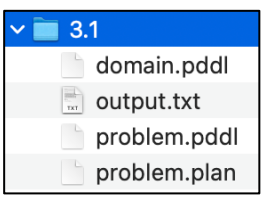
- **src**: contiene i file java riguardanti l'implementazione dell'euristica e degli algoritmi di ricerca.
- **pddl**: contiene i file .pddl riguardanti la modellazione del dominio e delle istanze.
- **lib**: contiene la libreria PDDL4J in formato .jar



Cartella *task3*: suddivisa in due sottocartelle



- **3.1**: contiene i file .pddl modellati per l'istanza temporale, e il file .plan generato dal planner LPG-td



- **3.2**: contiene il contenuto del problema trattato nell'omonima sezione 3.2, diviso nelle seguenti sottocartelle:
    - **launch**: che contiene il file commands e il file di launch .py
    - **pddl**: che contiene i file .pddl precedentemente descritti
    - **src**: che contiene le classi C++ corrispondenti alle fake action